نشریه علمی "پدافند الکترونیکی و سایبری"
سال هفتم، شماره ۲، تابستان ۱۳۹۸، ص ۶۷-۵۳

یک سامانه نرم‌افزاری برای شبیه‌سازی مقیاس‌پذیر انتشار بدافزارها در شبکه‌های رایانه‌ای

عرفان قناد توکلی[۱]، محمد عبداللهی ازگمی[۲*]
۱- کارشناس ارشد، ۲- دانشیار، دانشکده مهندسی کامپیوتر، دانشگاه علم و صنعت ایران، تهران
(دریافت: ۹۷/۰۲/۱۲، پذیرش: ۹۷/۰۷/۲۱)

چکیده

امروزه، وجود روش‌ها و ابزارهای مدل‌سازی پدیده‌های انتشاری همه‌گیر، یک نیاز جدی و پراهمیت است. شناخت این پدیده‌ها و کشف راه حل‌های راهبردی در موقعیت‌های گوناگون بسیار اهمیت دارد. به دلیل تعداد بالای عامل‌ها و رفتار پیچیده آن‌ها، امکان درک و توصیف آن‌ها به شکل نظری بسیار سخت و یا غیرممکن است. بنابراین، جایگاه شبیه‌سازی رفتار این پدیده‌ها بسیار حائز اهمیت است. لذا توسعه یک محیط شبیه‌سازی که بتواند این نیاز را تا حدی برطرف کند اهمیت بالایی دارد و می‌تواند در قسمت‌ها و حوزه‌های مختلف به‌کار گرفته شود. در این مقاله یک سامانه نرم‌افزاری برای شبیه‌سازی رایانه‌ای پدیده‌های انتشاری، به‌ویژه انتشار بدافزارها در شبکه‌های رایانه‌ای، معرفی شده است. این سامانه توانایی این را دارد که تعداد بالایی از عامل‌ها را با مدل‌های انتشاری و پارامترهای مختلف، شبیه‌سازی کرده و نتایج گرافیکی و آماری آن را به شکل پویا نشان دهد. معماری توزیع‌شده، استفاده از داده‌های واقعی، امکان استفاده از انواع مدل‌های انتشاری و تصویرسازی پویای نتایج شبیه‌سازی از جمله نوآوری‌های طرح پیشنهادی است. با انجام شبیه‌سازی‌های مختلف با استفاده از این سامانه، رابطه بین چگالی آلودگی عامل‌ها در طول زمان با پارامترهای شبیه‌سازی بررسی و استخراج شده است. مدل انتشاری، احتمال آلودگی، احتمال وجود آسیب‌پذیری از جمله این پارامترها است. همچنین در پایان، درستی نتایج حاصل از شبیه‌سازی با مقایسه نتایج حاصل از مدل‌های تحلیلی مورد بررسی قرار گرفته است.

کلید واژه‌ها: مدل‌سازی پدیده‌های انتشاری، انتشار بدافزارها، محیط شبیه‌سازی، شبیه‌سازی توزیع‌شده

۱- مقدمه

از جمله مسائلی که امروزه کشورهای جهان و از جمله کشور ما با آن روبرو است، رفتارهای همه‌گیر[۱] و به‌طور خاص انتشار بدافزار و یا بیماری در سطوح کلان و در مقیاس وسیع[۲] است. دانش در این مورد کمک به‌سزایی در مورد راهبردهای کشور در خصوص راه‌کارهای هجومی و دفاعی خواهد داشت. لذا تحلیل، طراحی و ساخت یک شبیه‌ساز که این نیازهای کشور را برطرف کند مسئله و چالش بزرگی است. این چالش در قسمت‌های مختلفی نمود دارد که از جمله آن می‌توان به موارد زیر اشاره داشت:

- تعداد بالا و مقیاس وسیع عامل‌ها[۳] در این شبیه‌سازی‌ها باعث نیاز به توان پردازشی بیشتر و همچنین مقدار حافظه مصرفی بسیار بالا است. این مقدار آن‌قدر می‌تواند بالا باشد که دیگر امکان مدیریت و اجرای آن بر روی یک ماشین تنها امکان‌پذیر نباشد.
- پیچیدگی رفتار پدیده‌های انتشاری به‌طوری که با نگاه کلی و از سطح بالا امکان شناخت رفتار میسر نیست.
- وجود جزئیات فراوان در این پدیده‌ها امکان استفاده از راه حل‌های مبتنی‌بر ریاضیات را بسیار دشوار کرده است. به‌طوری که در صورت استفاده از این روش‌ها باید ساده‌سازی‌ها و یکسان‌سازی‌های زیادی انجام داد که سبب کاهش دقت و کیفیت مدل محاسبه‌شده خواهد شد.
- به دلیل تعداد بالای عامل و موجودیت در سامانه، احتمال رخ‌دادن گلوگاه به شدت بالا است. لذا روش ارائه‌شده باید طوری باشد که از ایجاد گلوگاه در قسمت‌های مختلف از جمله شبکه، توان پردازشی و حافظه جلوگیری نماید.
- به دلیل تنوع در مدل‌های انتشاری و ادبیات‌های مختلف در این زمینه، راه حل ارائه‌شده باید به اندازه کافی انعطاف‌پذیر باشد که این تغییر در مدل انتشار را مدیریت کند.

بسیاری از پدیده‌های موجود در جهان با وجود اهمیت بسیار زیاد برای انسان‌ها، دامنه همه‌گیری کمی دارند. یعنی تعداد نفراتی که درگیر آن مشکل و مسئله هستند قابل توجه نیست. لذا علی‌رغم اهمیت بالا و چه‌بسا خطرناک بودن موضوع، با توجه به این‌که دامنه درگیری پایین است، نمی‌توان آن را یک چالش

*رایانامه نویسنده مسئول: azgomi@iust.ac.ir

1- Epidemic
2- Large-Scale
3- agents

بزرگ دانست. اما گاهی اوقات پدیده مورد بررسی یک پدیده همه‌گیر است. پدیده‌هایی که در شروع فعالیت شاید تعداد کمی از جمعیت را تحت تأثیر قرار دهند، اما با صرف زمان اندک، به دلیل ماهیت تصاعدی که در این پدیده‌ها وجود دارد، باعث درگیر شدن تمام جمعیت می‌شوند.

این پدیده می‌تواند جلوه‌های مختلفی داشته باشد، شیوع بیماری، انتشار شایعه و یا حتی انتشار بدافزار در تمامی این موارد نکته مهم این است که اگر اقدامی درخور و مناسب برای جلوگیری از انتشار رخ ندهد، صرف مدت‌زمان کمی، تمامی عامل‌ها تحت تأثیر قرار می‌گیرند.

بنابراین، شناخت این پدیده‌ها و کشف راه حل‌های راهبردی در موقعیت حمله و دفاع بسیار اهمیت دارد. جایگاه شبیه‌سازی در این بین بسیار پررنگ است، چراکه به دلیل تعداد بالای عامل‌ها و رفتار پیچیده آن‌ها، امکان درک و توصیف رفتار آن‌ها به شکل نظری بسیار سخت و یا غیرممکن است. بنابراین، توسعه یک محیط شبیه‌سازی که بتواند این نیاز را تا حدی برطرف کند اهمیت بالایی دارد و می‌تواند در قسمت‌ها و حوزه‌های مختلف به‌کار گرفته شود.

هدف اصلی پژوهشی که نتایج آن در این مقاله ارائه می‌شود، طراحی یک معماری مناسب برای ساخت یک محیط برای شبیه‌سازی مقیاس‌پذیر انتشار بدافزارها و یا هر پدیده انتشاری همه‌گیر است. برای ارزیابی طرح مورد نظر، معماری پیشنهادی، پیاده‌سازی شده و آزمون‌ها و آزمایش‌های مختلفی روی آن انجام شده و نتایج آن در این مقاله ارائه می‌شود. با کمک این محیط ساخته‌شده، می‌توان بررسی‌ها و تحلیل‌های مختلفی روی موضوعات مختلف مرتبط با پدیده‌های انتشاری و به‌طورخاص انتشار بدافزارها داشت و به کمک نتایج آن می‌توان به تصمیمات مهمی رسید.

در ادامه مقاله و در بخش ۲، در مورد کارهای مرتبط این حوزه صحبت خواهد شد. در بخش ۳، معماری پیشنهادی برای شبیه‌سازی مورد نظر، در بخش ۴، جزئیات فرآیند شبیه‌سازی، در بخش ۵ نمودار کلاس معماری ارائه‌شده، در بخش ۶ گزارش پیاده‌سازی معماری مورد نظر به منظور انجام آزمایش‌های مختلف و در بخش ۷ام با اجرای آزمایش‌های مختلف معماری پیشنهادی مورد ارزیابی قرار می‌گیرد. در بخش ۸ نیز به‌عنوان آخرین قسمت، نتیجه‌گیری و کارهای آینده بیان می‌شود.

۲- کارهای مرتبط

یکی از مواردی که شبیه‌سازی در آن از اهمیت بالایی برخوردار است، شبیه‌سازی رفتار و نحوه انتشار بدافزارها در شبکه‌های رایانه‌ای است. نوع انتشار واقعی و طبیعت انتشار بدافزار در شبکه‌های با مقیاس بالا از روی آسیب‌پذیری‌های نرم‌افزاری که توسط بدافزار مورد استفاده قرار می‌گیرند به‌دست می‌آید. هنگامی‌که در مورد تأثیرات امنیتی و اقتصادی انتشار بدافزار صحبت می‌کنیم، لزوماً نیازمند آن هستیم که شبکه‌های بسیار بزرگ و پیچیده را در نظر بگیریم. این شبکه‌ها می‌توانند زیرساخت‌های ارتباطی کشور، رابطه دوستی بین آدم‌ها و یا رفت و آمد بین آن‌ها باشد.

انتشار بدافزارها محور اساسی این مقاله است. فارغ از نوع بدافزار، خواه ویروس باشد چه اسب تراوا، چه جاسوس‌افزار باشد و چه باج‌گیرافزار، در هر صورت هر بدافزاری به‌طور کلی به روش یا روش‌هایی سعی در انتشار خود بر روی میزبان‌های دیگر دارند و به این طریق به شکل بسیار گسترده بر روی کامپیوترهای شبکه منتشر می‌شوند. انتشار بدافزارها را می‌توان به انتشار بیماری‌های واگیردار تشبیه کرد. زمانی که عامل بیماری‌زا شخص اول را مبتلا می‌کند، در صورت وجود شرایط مناسب، این عامل اشخاص دیگری را نیز بیمار می‌کند. بدافزارها نیز تقریباً به این‌گونه رفتار کرده و با توجه به نوع، برنامه و هدف خود سعی در انتشار خود دارند. تا امروز مدل‌های زیادی برای انتشار پدیده‌های همه‌گیر ایجاد شده است از جمله مدل SI[1]، SIS[2] و یا SIR[3]. از این مدل‌ها می‌توان در حوزه‌های گوناگون ازجمله انتشار بدافزارها استفاده کرد [۱].

در دهه گذشته شبیه‌سازی عامل-مبنا[4]، با موفقیت بر روی موضوعات مختلفی از علوم، نظیر علوم تجربی، میکروبی، جامعه‌شناسی و مهندسی و غیره اعمال شده است. با این وجود برای ادامه این موفقیت در سال‌های اخیر نیازمند افزایش حجم شبیه‌سازی‌ها، ترکیب دامنه‌ها و وسعت آن‌ها هستیم. این افزایش وسعت نیازمند توان بیشتر محاسباتی، زمان و هزینه بیشتر است. لذا طراحی و تولید بسترهایی برای شبیه‌سازی‌هایی با کارایی و مقیاس بزرگ بسیار حائز اهمیت است [۲].

Repast HPC یکی از این سکوها[5] است که قاعده اجرای موازی، توزیعی و همچنین اجرای عامل-مبنا را به‌خوبی پیاده‌سازی کرده است. این سکو قادر است تعداد زیادی مدل فردی را بر روی یک سامانه توزیع‌شده به‌خوبی اجرا کند. به‌عنوان نمونه، پخش شایعه می‌تواند مورد بررسی قرار گیرد. این کاربرد، پخش یک شایعه در بین یک جمعیت که مانند یک شبکه هستند

1- Susceptible-Infectious
2- Susceptible-Infectious-Susceptible
3- Susceptible-Infectious-Recovered
4- Agent-Based
5- Platform

را شبیه‌سازی می‌کند. این شبیه‌سازی در گام اول تعدادی از گره‌ها را به‌عنوان مبدأ شایعه در نظر می‌گیرد و سپس در هر تکرار، هر گره تعدادی از همسایه‌های خود را که تاکنون این شایعه به گوش آن‌ها نرسیده است را خبر کرده و آن‌ها را نیز به‌عنوان ارسال‌کننده جدید معرفی کرده و این چرخه ادامه پیدا می‌کند. ساختار شبکه در این شبیه‌سازی برای شباهت به مدل‌های واقعی، از مدل KE استفاده می‌کند. این مدل شبکه‌ای می‌سازد که اصطلاحاً به آن درجه قانون توان[1] می‌گویند [۳].

شبیه‌ساز حملات سایبری توزیع‌شده[2] یک سامانه برای شبیه‌سازی حملات سایبری بر اساس استاندارد HLA است. این سامانه به محققین کمک می‌کند که شبکه و تجهیزات جانبی آن را شبیه‌سازی و انواع حملات سایبری را بر روی آن شبیه‌سازی کنند. مزیت DCAS علاوه بر قدرت محاسباتی و کارایی بالا، مقیاس‌بزرگ بودن، توزیع‌پذیری، استفاده از HLA و رابط گرافیکی آن است که کمک شایانی به طراحی شبکه و مؤلفه‌های دیگر کرده است [۴].

GLEAM[3] یک ابزار شبیه‌سازی است که برای شبیه‌سازی انتشار بیماری انسان با یک رابط کاربری مناسب، به‌کار می‌رود. این ابزار می‌تواند انواع مدل‌های انتشار بیماری‌های همه‌گیر را شبیه‌سازی، و آمار و ارقام دقیق آن را گزارش دهد. یکی دیگر از ویژگی‌های خیلی کارآمد این ابزار وجود داده‌های واقعی، رفتار طبیعی، بر اساس جغرافیای طبیعی همراه با جزئیات گرافیکی است.

GLEAM از سه بخش تشکیل شده است، سرویس‌گیرنده، وکیل[4] و سرویس‌دهنده شبیه‌سازی. این در حالی است که فقط بخش سرویس‌گیرنده در رایانه شخصی نصب می‌شود و دو قسمت دیگر در میزبان‌های منتشرکننده قرار دارند و توسعه‌دهنده هیچ دسترسی به آن‌ها ندارد. ساختار پایه‌ای GLEAM از سه لایه تشکیل شده است، لایه مدل همه‌گیری[5]، لایه مدل حرکتی[6] و لایه مدل جمعیت[7]. لایه مدل شیوع توسط کاربر و در سرویس‌گیرنده طراحی و یا هر نوع مدل همه‌گیری رایج چون SI، SIS، و SIR انتخاب می‌شود. در لایه حرکتی، جمعیت موجود در مدل می‌تواند بر اساس دو نوع کوتاه-برد و بلند-برد در جغرافیا حرکت کنند. حرکت کوتاه برد به‌دلیل ارتباطات انسانی و حرکات بلند-برد به دلایلی چون سفر با هواپیما و یا قطار اتفاق می‌افتد.

1- Power-Law Degree
2- DCAS: Distributed Cyber Attack Simulation
3- Global Epidemic Mobility Model
4- Proxy
5- Epidemic Model
6- Mobility Layer
7- Population Data

نکته جالب توجه این است که اطلاعات پرواز دقیقاً بر اساس اطلاعات واقعی استخراج‌شده از پایگاه داده IATA و OAG است. در این اطلاعات حتی نوع هواپیما، زمان سفر و تعداد صندلی‌ها نیز در نظر گرفته شده است. اما لایه سوم در مورد جمعیت است، رویکرد این سامانه به این شکل است که مناطق قابل سکونت کره زمین را به‌صورت شبکه-شبکه و در حقیقت جدول‌بندی شده تقسیم کرده و هرکدام را یک زیرمجموعه از جمعیت در نظر گرفته است. این اطلاعات نیز به‌صورت واقعی و از پایگاه داده SEDAC دانشگاه کلمبیا استخراج شده است. با استفاده از این پایگاه داده، GLEAM نقاط زمین را به سلول‌هایی به عرض و طول ۲۵ کیلومتر تقسیم کرده است. حال با توجه به مدل همه‌گیری طراحی‌شده و اطلاعات مربوط به جمعیت، موقعیت آن‌ها و سفرهای آن‌ها، در ابتدا کافی است که شبیه‌سازی مقداردهی اولیه شود. مقداردهی اولیه شامل تعداد بیمارهای اولیه، موقعیت آن‌ها، کشور شیوع کننده، تنظیم متغیرهای مدل، نظیر نرخ احتمال انتقال بیماری و یا بهبود است. سپس کافی است که شبیه‌سازی را اجرا کرده و نتایج را به‌صورت پویا مشاهده نماییم [۵].

از نرم‌افزار NetLogo نیز به‌عنوان یکی دیگر از ابزارهای کارآمد در این حوزه می‌توان نام برد. این نرم‌افزار یک محیط مدل‌سازی چندعاملی قابل برنامه‌نویسی است [۶]. این نرم‌افزار توسط محققان زیادی مورد استفاده قرارمی‌گیرد و مدل‌های زیادی برای آن توسعه یافته است. از جمله آن‌ها می‌توان به کتابخانه "ویروس" اشاره کرد که در آن انتشار یک ویروس بیماری‌زا در جمعیت انسان‌ها مورد شبیه‌سازی قرار می‌گیرد [۷]. نکته بسیار حائز اهمیت این است که این نرم‌افزار دارای قابلیت‌های عمومی‌تر بوده و به‌طور اختصاصی روی حوزه خاصی فعالیت ندارد. همچنین باید توجه داشت که این نرم‌افزار دارای یک معماری ساده مبتنی بر کلاینت است و امکان شبیه‌سازی گره‌ها و ارتباطات بالا اصلاً وجود ندارد.

نوع دیگری از کارهای مرتبط با این زمینه، استفاده از مدل‌های تحلیلی و حل ریاضی به‌جای شبیه‌سازی‌های نرم‌افزاری است. این‌گونه کارها به‌عنوان یک دید سطح بالا و کلی روی موضوع بسیار اهمیت دارد اما جایگاه یک سامانه شبیه‌سازی بر اساس داده‌های واقعی کنار آن اکیداً توصیه می‌شود. حسینی و همکارانش [۸] در پژوهش خود به مدل‌سازی انتشار بدافزارها در شبکه‌های بی‌مقیاس[8] وزن‌دار پرداخته است و نقش تنوع نرم‌افزاری[9] در انتشار را بررسی کرده است.

8- Scale-Free Networks
9- Software Diversity

در این مقاله، بر اساس مدل همه‌گیری SEIRS[1] پویایی انتشار بدافزار در شبکه بی‌مقیاس وزن‌دار مورد مطالعه قرار گرفته است. همچنین تاثیر تنوع نرم‌افزاری به‌عنوان یک راهبرد دفاعی در مدل‌سازی انتشار بدافزار در نظر گرفته‌شده و بسته‌های نرم‌افزاری متنوع به گره‌های شبکه تخصیص داده شده است تا از انتشار همه‌گیر در شبکه بی‌مقیاس وزن دار جلوگیری شود. با استفاده از حل تحلیلی و شبیه‌سازی‌های عددی، تاثیر پارامترهای مختلف روی فرآیند انتشار بدافزار مورد مطالعه قرار گرفته است. همچنین نشان داده شده است که گره‌ها با درجه بالا و قدرت بالاتر، آلودگی بدافزار را سریع‌تر منتشر می‌کنند.

باید اشاره داشت که هیچ‌کدام از سامانه‌های ذکرشده به‌طور کامل جواب‌گوی نیاز اصلی ما نیست. سامانه GLEAM تنها برای شبیه‌سازی انتشار بیماری مورد استفاده قرار می‌گیرد و امکان تغییر این مورد وجود ندارد. همچنین این سامانه امکان اضافه کردن و یا تغییر داده‌های مکانی و جغرافیایی و یا حتی حرکتی عامل‌ها را نداشته و این موارد تنها دست سازندگان بوده و برای مصرف‌کننده قابل تغییر و یا حتی مشاهده نیست. همچنین قسمت سرویس‌دهنده این سامانه متن‌باز نبوده و امکان ویرایش و مطالعه آن وجود ندارد. در مورد سامانه DCAS هم باید اشاره داشت که هدف این سامانه اساساً تحلیل تأثیر حملات سایبری بر روی کارایی شبکه است و تمرکز کافی در مورد انتشار بدافزار و مدل‌های انتشاری ندارد. نوع کارهایی که بر اساس حل تحلیلی و مدل ریاضی به بررسی موضوع پرداخته‌اند نیز اساساً با هدف پژوهش تفاوت دارد. سامانه REPAST نیز که تنها به‌عنوان نمونه‌ای از یک شبیه‌سازی با کارایی بالا آورده شده است و در مورد مدل‌های انتشاری و بحث شبیه‌سازی بدافزار مناسب نیست.

۳- معماری پیشنهادی برای شبیه‌سازی مقیاس‌پذیر انتشار بدافزار

۳-۱- نیازمندی‌های سامانه

برای طراحی و اجرای هر نرم‌افزار، اولین قدم تحلیل و بررسی نیازمندی‌ها است. به همین دلیل در قدم اول با توجه به شرح مسئله و انگیزه پژوهش که در قسمت‌های قبلی بیان شد و با الهام از کارهای مرتبطی که در این زمینه انجام شده است و با عنایت به نظرات و مشورت‌های افراد صاحب‌نظر، نیازمندی‌های اساسی برای این سامانه به شرح زیر به‌دست آمده است.

۱) **مقیاس‌وسیع بودن**[2]: مهم‌ترین مؤلفه در پدیده‌های همه‌گیر، وجود تعداد بسیار زیاد عامل در آن‌ها است و اصولاً بررسی یک پدیده همه‌گیر در مقیاس کوچک، کاری بیهوده بوده و نتایج آن به هیچ‌وجه کارآمد نیست. از این‌رو باید روشی طراحی شود که بتواند تعداد زیادی عامل را در نظر بگیرد.

۲) **مقیاس‌پذیری**[3]: به‌طور رسمی یک سامانه را زمانی مقیاس‌پذیر می‌نامند که آن سامانه بتواند افزایش بار کاری را تحمل و مدیریت کند. یعنی اگر با تعداد خاصی عامل توانست کار کند، با تعداد بیشتری هم بتواند کار کند.

۳) **زمان اجرا**: زمان اجرای شبیه‌سازی، پارامتر بسیار مهمی است. اگر زمان اجرای هر شبیه‌سازی چندین روز یا حتی چندین ساعت باشد، استخراج نتایج کاربردی و استفاده از نتایج شبیه‌سازی کاری دشوار خواهد بود. زیرا اساس این سامانه با تغییر در پارامترها و مشاهده نتیجه معنا دارد. اگر هر اجرای آن زمان زیادی طول بکشد، این مورد عملاً باعث کاهش کیفیت نتیجه خواهد بود.

۴) **نمایش نتایج**: حتماً باید نتایج شبیه‌سازی به روشی درست و ملموس در اختیار کاربر نهایی قرار بگیرد. ارائه آمارهای تحلیلی و یا مشاهده وضعیت کلی عامل‌ها و نحوه اتصال آن‌ها به یکدیگر اهمیت بالایی دارد.

۵) **قابلیت استفاده مجدد**[4]: یکی از مهم‌ترین مسائلی که باید در طراحی این سامانه در نظر گرفته شود، جداسازی مؤلفه‌های مختلف سامانه است. مدل انتشاری و یا مؤلفه نمایش نباید به شکلی پیاده‌سازی شود که به بدنه اصلی شبیه‌سازی لطمه وارد کند. تغییر داده‌های ورودی و یا تغییر رابط شبکه و فنّاوری ارسال پیام نیز باید همین‌گونه باشد.

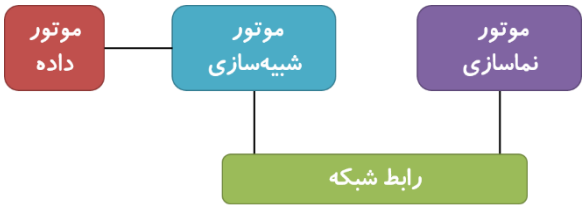
۳-۲- معماری سامانه

مهم‌ترین مسئله‌ای که در طراحی این محیط باید به آن دقت شود، معماری استفاده‌شده در آن است. معماری نرم‌افزار نقش تعیین‌کننده و بسیار مهمی در سرعت اجرای برنامه، کیفیت، دقت، انعطاف‌پذیری، قابلیت استفاده مجدد و مقیاس‌پذیری برنامه دارد. از این‌رو، طراحی یک معماری صحیح که پاسخگوی نیاز مطرح‌شده باشد، مهم‌ترین قسمت این مقاله است.

همان‌طور که در شکل (۱) مشاهده می‌شود طرح مفهومی معماری پیشنهادی از چهار مؤلفه اصلی تشکیل شده است. هرکدام از این مؤلفه‌ها مسئولیت بخشی از سامانه را بر عهده داشته و فقدان هرکدام می‌تواند بخشی از عملکرد سامانه را مختل

1- Susceptible-Exposed-Infected-Recovered-Susceptible
2- Large Scale
3- Scalablity
4- Reusabilty

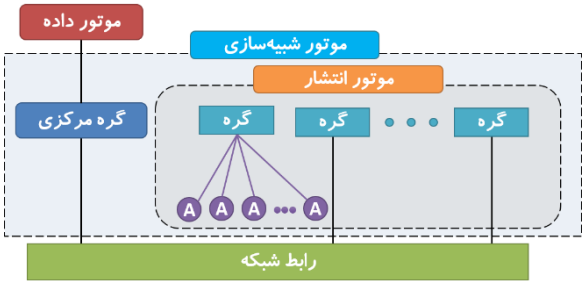
کند. باید توجه داشت که در طراحی این سامانه تمامی نیازمندی‌های مطرح‌شده در بخش اول در نظر گرفته شده است. در ادامه هر بخش به‌طور کلی توضیح داده می‌شود.



شکل(۱): مدل مفهومی معماری ارائه‌شده، مؤلفه‌های سازنده و ارتباط بین آن‌ها

۳-۳- موتور شبیه‌سازی

این بخش از سامانه، در حقیقت هسته مرکزی معماری بوده و کار اصلی شبیه‌سازی را انجام می‌دهد. بخشی که مسئولیت خواندن داده‌های پردازش‌شده، ارسال آن‌ها برای گره‌های محاسباتی و در حقیقت شبیه‌سازی پدیده مورد نظر است. این موضوع که بدافزار چگونه انتشار یابد؟ مدل انتشاری آن چیست؟ و یا سرعت و میزان انتشار بدافزار روی میزبان‌ها چگونه است؟ همه این موارد در این بخش مدیریت شده و از اهمیت بسیار بالایی برخوردار است. این مؤلفه در شکل (۲) نشان داده شده است.



شکل(۲): معماری موتور شبیه‌سازی

- **گره مرکزی**[1]

هسته اصلی و شروع‌کننده فرایند شبیه‌سازی این مؤلفه است. از نظر پردازشی این گره نیازمند پردازش سنگینی نیست و صرفاً یک شروع‌کننده بوده و وظیفه تنظیم کردن پارامترها و ویژگی‌های موتور انتشار را دارد. با شروع فرایند شبیه‌سازی، این قسمت با بخش موتور داده‌ها ارتباط برقرار کرده و داده‌های اولیه مورد نیاز برای اجرای شبیه‌سازی را به‌دست می‌آورد. در مرحله بعد این داده‌ها همراه با پارامترهای شبیه‌سازی تحویل گره‌های محاسباتی می‌شود و به هر گره محاسباتی اعلام می‌شود که مسئولیت شبیه‌سازی کدام عامل‌ها را بر عهده دارد. پارامترهایی که از طریق گره مرکزی به موتور انتشاری ارسال می‌شود به‌صورت زیر است:

- **زمان تلاش برای ارتباط یک عامل با یک عامل دیگر**: یک عامل در هر چند ثانیه با یکی از عامل‌های متصل به خود ایجاد ارتباط می‌کند؟
- **زمان تلاش برای ترمیم**: یک عامل هر چند ثانیه تلاش می‌کند تا وضعیت خود را از حالت آلودگی به حالت معمولی تبدیل کند؟
- **احتمال آلوده شدن**: به فرض ایجاد یک ارتباط آلوده و به فرض وجود آسیب‌پذیری در عامل، چقدر احتمال دارد که عامل میزبان این آلودگی را دریافت کند؟
- **احتمال داشتن آسیب‌پذیری**: چقدر احتمال دارد، عامل یاد شده، آسیب‌پذیری موجود در پیام را داشته باشد؟
- **احتمال ترمیم شدن**: در هر یک از تلاش‌های عامل برای ترمیم، چقدر احتمال موفقیت وجود دارد؟
- **مدل انتشار**: گره مرکزی هنگام فراخوانی موتور انتشار نوع مدل انتشار را انتخاب می‌کند. این مدل می‌تواند یکی از مدل‌های SI، SIR، SIS و غیره باشد.

- **عامل**

عامل‌ها در حقیقت یک میزبان مدل انتشاری هستند. به‌عبارت دیگر به فرض استفاده از مدل SIR هر یک از این عامل‌ها حالتی مشخص مثل مستعد آلودگی، آلوده و یا ترمیم‌شده دارند. هر عامل مشخصات و ویژگی‌هایی دارد که از طریق موتور داده مقداردهی شده و از طریق گره مرکزی به گره محاسباتی واگذار می‌شود و گره محاسباتی عاملی با مشخصات یاد شده ایجاد می‌کند. از این به بعد این عامل به طور مستقل رفتار کرده و خود چرخه حیات خود را طی می‌کند. بنابراین، سامانه یک رویکرد چندعاملی داشته و هر عامل به طور مجزا و مستقل رفتار می‌کند. عامل‌ها که نماینده یک سامانه رایانه‌ای با قابلیت آلودگی هستند، به صورت یک ماشین حالت متناهی مدل می‌شوند که با توجه به پارامترهای انتشار از حالتی به حالت دیگر می‌روند. هر عامل دارای مشخصات زیر است:

۱) **شناسه**: عدد یکتایی که توسط موتور داده، هنگام آماده‌سازی داده خام، تولید و به گره مرکزی ارسال می‌شود. از طریق همین شناسه است که گره مرکزی تصمیم می‌گیرد مسئولیت این عامل را به کدام گره محاسباتی بسپارد و به همین طریق، گره محاسباتی متوجه می‌شود که یک عامل عضو کدام گره است.

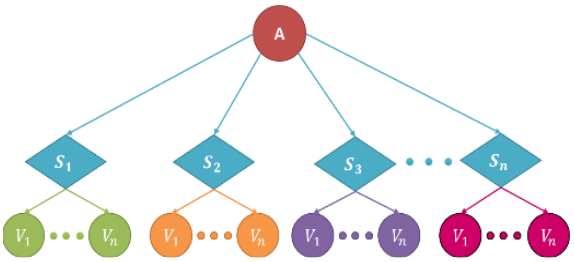
۲) **گره محاسباتی**: گره محاسباتی که مسئولیت این عامل را بر عهده دارد.

1- Master Node

۳) **مدل انتشاری**: مشخص کننده مدل انتشاری استفاده شده در این عامل است. هر عامل، مسئول مدیریت و اجرای ماشین متناهی متناسب با مدل انتشاری را دارد.

۴) **عامل‌های متصل**: عامل‌هایی که به‌طور مستقیم امکان ارتباط با آن‌ها وجود دارد و به نوعی بین آن‌ها ارسال پیام وجود دارد.

۵) **آسیب‌پذیری‌ها**[1]: هر عامل (A) مجموعه‌ای از سامانه‌های نرم‌افزاری (S_1 تا S_n) دارد که هر کدام از آن‌ها دارای آسیب‌پذیری‌هایی (V_1 تا V_n) هستند. مجموعه این موارد، آسیب‌پذیری‌های یک عامل را مشخص می‌کند. در شکل (۳) این مدل به تصویر کشیده شده است.



شکل (۳): درخت آسیب‌پذیری‌های یک عامل

- **گره محاسباتی**

گره محاسباتی یا به‌اختصار گره، مسئول و محاسبه‌گر فرایند شبیه‌سازی است. در درون هر گره تعداد زیادی عامل وجود دارد. گره محاسباتی تمامی این عامل‌ها را شناخته و عملیات انتشار بین آن‌ها را محاسبه می‌کند. گره‌ها عامل اصلی در فرایند شبیه‌سازی توزیع‌شده هستند. هر گره در حقیقت یک پردازه است که می‌تواند روی هر رایانه‌ای اجرا شود. بنابراین، کافی است هر گره را فراخوانی کرده و یک عدد یکتا به‌عنوان شناسه گره اعلام کنیم. سپس گره از طریق شبکه، خود را به گره مرکزی معرفی می‌کند و به همین طریق گره‌های زیادی بر روی گره مرکزی ثبت نام می‌کنند. سپس گره مرکزی با توجه به تعداد عامل‌ها و تعداد گره‌ها یک فرایند متعادل‌سازی بار انجام داده و با توجه به قدرت سخت‌افزاری هر گره، تعدادی از عامل‌ها را برای شبیه‌سازی در اختیار این گره قرار می‌دهد.

موتور انتشار تنها از این گره‌ها آگاهی دارد و این گره‌ها هستند که اطلاعات تمامی عامل‌ها را در اختیار دارند. گره‌ها وظیفه دارند هر تغییری که در عامل‌های خود به وجود می‌آید را به اطلاع موتور نماسازی رسانده و همچنین وظیفه ارسال پیام یک عامل به عامل دیگر بر عهده گره محاسباتی است. بنابراین، یک گره محاسباتی دو پارامتر اصلی دارد: یک شناسه یکتا و فهرستی از عامل‌های تحت پوشش.

- **نحوه تخصیص شناسه و یافتن گره مسئول یک عامل**

یکی از چالش‌هایی که در طراحی گره‌ها وجود دارد، شناسه یکتای آن‌ها است. این شناسه از آن‌جایی که قرار است در رابط شبکه به‌عنوان کلید عمل کند؛ باید یکتا باشد. علاوه‌بر این، هر گره باید به‌نحوی شناسه گره‌های دیگر را نیز بداند. زیرا ممکن است یک عامل خواستار ارتباط و ارسال پیام به عاملی باشد که تحت پوشش این گره نیست. گره باید از طریق روشی شناسه گره مورد نظر را محاسبه کند و به وی پیام ارسال کند.

برای حل این مشکل، نحوه توزیع عامل‌ها میان گره‌ها را به به‌طوری انجام داده‌ایم که از روی شناسه هر عامل، شناسه گره قابل محاسبه باشد. شاید ساده‌ترین روش آن محاسبه باقی‌مانده باشد. به‌طور مثال اگر تعداد گره‌های محاسباتی را ثابت فرض کنیم و همچنین به تمامی عامل‌ها یک عدد ثابت تخصیص دهیم. در این صورت گره مرکزی می‌تواند عامل‌ها را به‌طور مساوی بین گره‌ها تقسیم کند. برای این کار کافی است شناسه گره مورد نظر برای هر عامل را از طریق رابطه (۱) محاسبه کند.

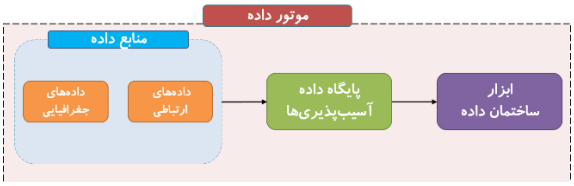
$$key = agentKey \,\%\, nodesCount \qquad (۱)$$

وقتی گره مرکزی عامل‌ها را به این طریق میان گره‌ها تقسیم کند، دقیقاً کافی است هر گره نیز از همین رابطه استفاده کند و شناسه گره مورد نظر خود را پیدا کند. تعداد تمامی گره‌ها را گره مرکزی می‌تواند در پیام اولیه خود به موتور انتشار ارسال کند.

۳-۴- موتور داده

این مؤلفه وظیفه تهیه، پردازش و تبدیل داده‌های خام به ساختمان داده مناسب برای فرایند شبیه‌سازی را بر عهده دارد. این داده‌ها به دو دسته تقسیم می‌شوند. داده‌های جغرافیایی و داده‌های ارتباطی. داده‌های جغرافیایی شامل مکان هر عامل بوده و از دو پارامتر طول و عرض جغرافیایی تشکیل شده‌اند. داده‌های ارتباطی نیز نمایانگر گرافی از عامل‌ها هستند. گرافی که گره‌های آن مشخص‌کننده عامل‌ها و یال‌های آن نشان‌گر وجود ارتباط بین دو عامل است. وقتی از وجود ارتباط بین دو عامل صحبت می‌شود می‌تواند معانی مختلفی در زمینه‌های مختلف داشته باشد. امکان پویش از طرق شبکه، ارسال پست الکترونیکی، ارسال پیام مستقیم، وجود ارتباط فامیلی یا دوستی بین دو عامل و غیره. فارغ از نوع این تفسیر، آنچه در انتشار پدیده‌های همه‌گیر مهم است، وجود خود ارتباط است. در شکل (۴) مؤلفه‌های سازنده و نحوه ارتباط آن‌ها آمده است.

1- Vulnerability



شکل (۴): موتور داده و مؤلفه‌های سازنده

- **پایگاه داده آسیب‌پذیری‌ها**

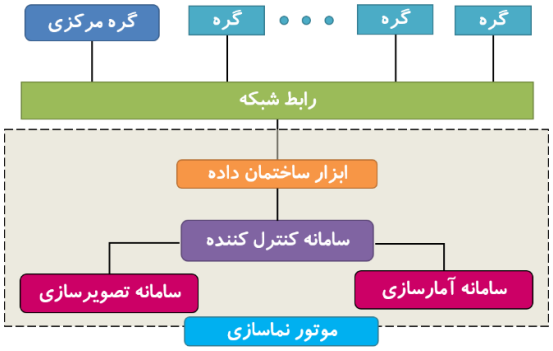
این پایگاه داده شامل اطلاعات نرم‌افزاری و آسیب‌پذیری‌های مورد استفاده برای مقداردهی اولیه عامل‌ها هستند.

- **ابزار ساختمان داده**

بر طبق قراردادی که مابین موتور شبیه‌سازی و موتور داده شده است، موتور داده موظف است داده‌های خام را به ساختار مناسب تبدیل کرده و آن را در اختیار موتور شبیه‌سازی قرار دهد.

۳-۵- موتور نماسازی[1]

موتور نماسازی وظیفه تصویرسازی و به نمایش گذاشتن نتایج شبیه‌سازی را داشته و از اهمیت بالایی برخوردار است. این قسمت هم می‌تواند به‌صورت متنی و هم به‌صورت تصویری باشد. همچنین انواع کارهای آماری در زمینه مربوطه می‌تواند انجام شود. روال کاری این قسمت از سامانه از دو فاز تشکیل شده است. فاز اول شامل دریافت اطلاعات ثابت و تغییر ناپذیر از گره مرکزی و فاز دوم شامل دریافت تغییرات حاصل از اجرای شبیه‌سازی از گره‌های محاسباتی است. با توجه به عدم تمرکز مقاله در این حوزه، در این قسمت تنها به معرفی کلی این بخش می‌پردازیم.



شکل (۵): موتور نماسازی و ارتباط اجزای سامانه با آن

همانطور که در شکل (۵) مشخص است، گره‌های محاسباتی و همچنین گره مرکزی از طریق شبکه اطلاعات خود را به موتور نماسازی ارسال می‌کنند. در ادامه مؤلفه ابزار ساختمان داده، این اطلاعات را به ساختمان داده مناسب تبدیل کرده و تحویل سامانه کنترل‌کننده می‌دهد. در پایان این سامانه با توجه به نوع اطلاعات و با توجه به زمان‌بندی‌های دیگر، اطلاعات لازم را به سامانه آمارسازی و یا سامانه تصویرسازی ارسال می‌کند.

1- Visualization

۳-۶- رابط شبکه

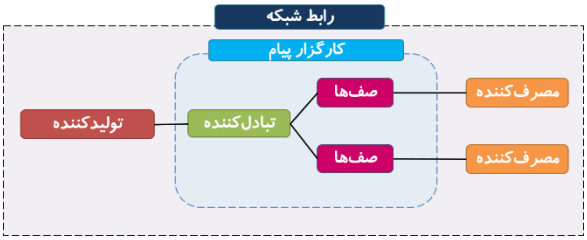
تبادل پیام بین دو مؤلفه که در یک ماشین فیزیکی قرار ندارند، از طریق رابط شبکه امکان‌پذیر است. زیرا اساس این معماری بر پایه اصول توزیع شده‌گی ساخته شده است. دلیل استفاده از این معماری این است که برای رسیدن به هدف مقیاس‌پذیری، تنها راه موجود، افزایش تعداد رایانه‌ها و سامانه‌های محاسبه‌گر است و اصولاً دیگر با سامانه‌های متمرکز دست‌یابی به اهداف گفته‌شده امکان‌پذیر نیست. معماری فعلی را شاید بتوان یک ترکیب از سامانه‌های غیر متمرکز[2] و سامانه‌های توزیع‌شده[3] دانست. زیرا ارتباط عامل‌ها با یکدیگر به شکل غیرمتمرکز ولی ارتباط گره‌های محاسباتی به شکل توزیعی است. این مفهوم در شکل (۷) بهتر نشان داده شده است.

هدف اصلی این قسمت، رساندن پیام یک فرستنده به دست مصرف‌کننده است. معماری ارسال پیام به‌طور کلی در شکل (۶) مشخص شده است. همان‌طور که مشخص است این موتور از چهار مؤلفه اساسی تشکیل شده است. تولیدکننده[4]، مصرف‌کننده[5]، صف‌ها[6] و تبادل‌کننده[7]. البته از نظر ساختاری می‌توان مجموعه صف‌ها و تبادل‌کننده‌ها را کارگزار پیام[8] نامید. این معماری به‌دلیل استفاده از میان‌افزار RabbitMQ استفاده شده است.

معماری RQ به این‌گونه است که فرستنده پیام خود را به یک تبادل‌کننده که در سرور قرار دارد، ارسال می‌کند. حال مصرف‌کننده برای دریافت پیام یک صف ساخته و این صف را اصطلاحاً بر روی تبادل‌کننده متصل[9] می‌کند. سرور پیام‌هایی که برای این صف آماده شده است را وارد آن می‌کند و مصرف‌کننده به‌ترتیب آن‌ها را برداشته و استفاده می‌کند. اتصال هر صف به تبادل‌کننده نیازمند یک کلید است. این کلید هنگام ارسال پیام به تبادل‌کننده استفاده می‌شود و در واقع گیرنده پیام را مشخص می‌کند. این کلید همان شناسه یکتای تولید شده برای هر گره

2- Decentralized
3- Distributed
4- Producer
5- Consumer
6- Queue
7- Exchange
8- Message Broker
9- Bind

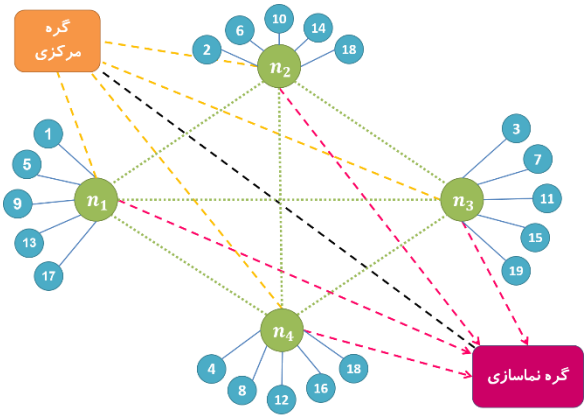
محاسباتی است که قبلاً اشاره شده بود. به این طریق گره‌های محاسباتی و یا گره مرکزی با دانستن شناسه گره‌های دیگر امکان ارسال پیام به آن‌ها را دارند.



شکل (۶): معماری ارسال پیام در مؤلفه شبکه [۹]

۴- جزئیات فرایند شبیه‌سازی انتشار

برای روشن شدن موضوع، نحوه ارتباط اجزاء سامانه را به طور دقیق شرح می‌دهیم. در ابتدا شکل شماره (۷) را در نظر بگیرید. در این شکل دایره‌های بزرگ n_1 تا n_4 به معنی گره‌های محاسباتی، دایره‌های کوچک ۱ تا ۲۰ به معنی عامل‌ها و دو مستطیل ارائه‌شده به ترتیب از بالا به پایین نماینده گره مرزی و گره نماسازی است. همچنین خطوط کامل به معنی ارتباط داخلی و خطوط نقطه‌چین به معنی ارتباط از طریق شبکه است.



شکل (۷): نحوه استقرار و ارتباط اجزای سامانه

همان‌طور که در شکل (۷) مشخص است، عامل‌ها به شکل غیر متمرکز به یکدیگر متصل هستند. یعنی هر عامل برای ارتباط و در حقیقت ارسال پیام به عامل دیگر، از طریق گره محاسباتی اقدام می‌کند. بخش دیگر تصویر مربوط به ارتباط گره‌های محاسباتی است. همان‌طور که مشخص است این گره‌ها به‌طور کاملاً توزیعی با یکدیگر ارتباط دارند. به‌عبارت دیگر سلسله مراتب آن‌ها یکسان است. فارغ از این‌که این گره‌ها در کدام کامپیوتر فیزیکی قرار دارند، هرکدام یک پردازه هستند و برای ارتباط با یکدیگر از رابط شبکه استفاده می‌کنند. همچنین تمامی آن‌ها امکان انتقال پیام با یکدیگر را داشته و هیچ دسته‌بندی خاصی وجود ندارد و آن‌ها به شکل نظیربه‌نظیر با یکدیگر ارتباط دارند. این معماری کمک می‌کند که یک گره مرکزی محل تجمع پیام‌های ارتباطی نشود و به همین دلیل هیچ گره‌ای یک گلوگاه نخواهد شد.

گره مرکزی از نظر ارتباطی، با گره‌های محاسباتی و با گره نماسازی در ارتباط است. گره نماسازی همان‌طور که قبلاً اشاره شد، وظیفه نمایش نتیجه شبیه‌سازی را بر عهده دارد و گره‌ها تنها تغییرات مشاهده شده در عامل‌های خود را به این گره اعلام می‌کنند. مثلاً اعلام می‌دارند، عامل شماره ۵ از گره n_1 آلوده شد. گره نمایش نیز یک ارتباط با گره مرکزی برای مقداردهی اولیه و دریافت اطلاعاتی چون تعداد عامل‌ها، تعداد گره‌ها، مکان آن‌ها و نحوه اتصالات آن‌ها دارد.

در این قسمت یک سناریو فرضی از روند اجرای شبیه‌سازی را بیان می‌کنیم. بعد از این‌که گره مرکزی از موتور داده، اطلاعات لازم را دریافت کرد و بعد از تقسیم‌بندی عامل‌ها میان گره‌های موجود در سامانه، اطلاعات اولیه و لازم برای نمایش را به گره نمایش ارسال کرده و همچنین عامل‌های تحت پوشش هر گره و همچنین تنظیمات کلی شبیه‌سازی را برای گره‌ها ارسال می‌کند. گره‌های محاسباتی آن‌ها را دریافت کرده و عامل‌های مورد نظر را تولید و فرایند شبیه‌سازی را آغاز می‌کنند. با آغاز فعالیت، هر عامل طبق پارامترهای تنظیم‌شده سعی در برقراری ارتباط به عامل‌های متصل به خود را دارد. عامل ارسال‌کننده با شناسه ۱، به گره محاسباتی خود یعنی n_1 اعلام می‌دارد که خواهان ارسال پیام به عامل با شناسه ۱۵ است. گره n_1 با توجه به شناسه مقصد تشخیص می‌دهد که آیا این عامل تحت پوشش خود است یا خیر. در صورتی که عامل تحت پوشش باشد، یک پیام داخلی و در غیر این صورت یک پیام تحت شبکه به گره مسئول عامل ارسال می‌کند. در این مثال با توجه به این‌که تعداد گره‌های محاسباتی ۴ بوده و شناسه عامل مقصد ۱۵ است، با توجه به توضیحات ارائه شده در قسمت‌های قبلی مقاله، از آن‌جایی که باقی‌مانده تقسیم ۱۵ بر ۴، عدد ۳ می‌شود که این عدد، شناسه گره n_3 در شبکه فرضی است. گره n_3 با دریافت این پیام، عامل مقصد، یعنی ۱۵ را پیدا کرده و پیام را به وی می‌رساند. حال با توجه به مدل انتشاری و پارامترهای شبیه‌سازی و وضعیت فعلی عامل، اگر این عامل آلوده شود، گره n_3 یک پیام مبنی بر آلوده شدن عامل ۱۵ به گره نمایش ارسال می‌کند. حال اگر عامل ۱۵ در وضعیت مستعد آلودگی قرار داشت، به وضعیت آلوده شده تغییر حالت داده و خود از این پس خواهان آلوده‌سازی عامل‌های متصل به خود است. توجه به این نکته الزامی است که زمانی یک عامل آلوده می‌شود که اولاً داری آسیب‌پذیری مورد نظر موجود در پیام باشد و ثانیاً عامل، محصول دفاعی خاصی برای مقابله با آن تهدید

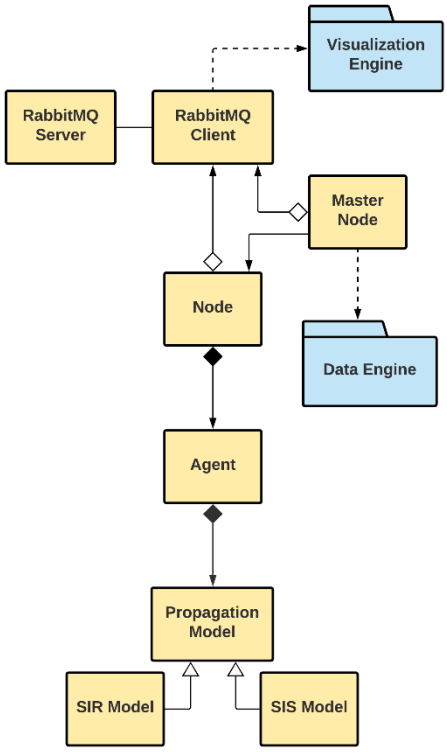
را نداشته باشد و ثالثاً از نظر مدل انتشاری، در وضعیت مستعد آلودگی باشد.

۵- نمودار کلاس

در این قسمت نمودار کلاس معماری ارائه خواهد شد. تمرکز اصلی روی طراحی این بخش، در بخش موتور شبیه‌سازی بوده و قسمت‌های جانبی به صورت یک بسته نشان داده شده است. همان‌طور که در تصویر (۸) قابل مشاهده است، کلاس‌های اصلی و همچنین نوع رابطه آن‌ها با یکدیگر نشان داده شده است.

۶- پیاده‌سازی شبیه‌سازی

گام اول برای ارزیابی این معماری، پیاده‌سازی یک سامانه مبتنی بر این معماری است. سامانه مقیاس‌پذیر انتشار بدافزارها یا به طور اختصار «سماب» یک محیط شبیه‌سازی توزیع شده است که دقیقاً بر اساس معماری ارائه شده در این مقاله پیاده‌سازی شده است. در ادامه این بخش، جزئیات پیاده‌سازی این سامانه به اختصار بیان می‌شود. برای پیاده‌سازی مؤلفه شبیه‌سازی این سامانه از زبان C# و چارچوب Net استفاده شده است. همچنین از میان‌افزار RabbitMQ به‌عنوان رابط شبکه و از موتور Unity به‌عنوان موتور نماسازی استفاده شده است.



شکل (۸): نمودار کلاس معماری

۶-۱- مشخصات سخت‌افزاری

برای اجرای این سامانه حداقل یک گره مرکزی، یک گره نماسازی، یک گره سرور و تعدادی گره محاسباتی نیاز است. به دلیل شرایط آزمایشگاهی و آزمون‌های اولیه، تمامی این پردازه‌ها بر روی یک سامانه اجرا گشته و نتایج بر روی آن به‌دست آمده است. در جدول (۱) مشخصات سخت‌افزاری سامانه آورده شده است.

جدول (۱): سامانه سخت‌افزاری دستگاه مورد آزمایش

نام قطعه	مدل
پردازنده مرکزی	Intel Core i7 4702MQ (2.2 GHz)
حافظه	6 GB DDR3
کارت گرافیک	NVIDIA GT 740 M (2GB Vram)

همچنین برای آزمایش معماری مبتنی بر سامانه‌های توزیعی، از ۴ گره محاسباتی استفاده شده است. بنابراین با اجرای کلی شبیه‌سازی، ۷ پردازه بر روی سامانه اجرا می‌شوند:

۱. گره مرکزی
۲. ۴ گره محاسباتی
۳. سرور RQ
۴. موتور یونیتی

۶-۲- آماده‌سازی داده

مجموعه داده‌هایی که برای اجرای کار و شبیه‌سازی از آن‌ها استفاده شد از وب‌سایت Snap [۱۰] تهیه شده است. این مجموعه داده از دو پرونده تشکیل شده است. پرونده اول مربوط به اطلاعات مکانی گره‌ها و پرونده دوم مربوط به وجود یال و در حقیقت ارتباط ما بین گره‌ها است. در پرونده‌های گراف همواره تفسیر ما از گره‌ها همان عامل‌های شبیه‌سازی و از وجود یال همان وجود ارتباط بین دو عامل است. مجموعه داده Brightkite [۱۰] یک مجموعه داده مبتنی بر مکان است که ۵۸۲۲۸ گره و ۲۱۴۰۷۸ یال در آن وجود دارد.

۶-۳- فرایند شبیه‌سازی

همان‌طور که قبلاً شرح داده شد، گره مرکزی، عامل‌های به‌دست آمده و پردازش‌شده را بین گره‌های محاسباتی تقسیم کرده و به نوعی یک تقسیم بار انجام می‌دهد. هرکدام از این گره‌های محاسباتی می‌تواند در یک رایانه مجزا اجرا شود و کافی است که بتواند به سرور سامانه متصل شود. بنابراین، برای شروع فرایند شبیه‌سازی لازم است از این پردازه به تعداد دلخواه اجرا کرده تا گره مرکزی آن را به‌عنوان یک گره محاسباتی بشناسد. با شروع پردازه گره محاسباتی، یک پیام حاوی شناسه گره برای گره مرکزی ارسال می‌شود. گره مرکزی بعد از دریافت تمامی این شناسه‌ها فرایند شبیه‌سازی را آغاز می‌کند. این کار با ارسال یک پیام اولیه که شامل پارامترهای شبیه‌سازی و همچنین عامل‌های

تحت پوشش هر گره محاسباتی است آغاز می‌شود. همچنین گره مرکزی در این پیام وضعیت اولیه هر عامل را نیز مشخص می‌کند.

در آزمایش‌های ما تمامی عامل‌هایی که شناسه آن‌ها مضرب ۵ است به‌طور پیش‌فرض در وضعیت آلوده قرار دارند. به این طریق حدود ۲۰ درصد از عامل‌ها در شروع فرایند شبیه‌سازی در وضعیت آلودگی قرار دارند. توجه به این نکته الزامی است که وجود آلودگی اولیه در این شبیه‌سازی الزامی است و اگر هیچ عاملی در ابتدا آلوده نباشد، هیچ‌گاه پدیده انتشار رخ نخواهد داد. گره محاسباتی بعد از دریافت پیام اولیه عامل‌های مورد نظر را ایجاد و مقداردهی کرده و هریک از عامل‌ها با توجه به پارامترهای شبیه‌سازی، مدل انتشاری خود را تنظیم کرده و فرایند انتشار را آغاز می‌کنند. در دوره‌های زمانی خاصی که قبلاً تنظیم شده است، عامل‌ها سعی در انتشار پیام آلوده بین عامل‌های متصل خود را دارند. بنابراین در هر دوره زمانی، عامل‌هایی که در وضعیت آلوده قرار دارند ابتدا یکی از عامل‌های متصل خود را انتخاب کرده، و سپس یک پیام آلوده مبتنی بر یک آلودگی (آلودگی که قبلاً با آن آلوده شده است) برای وی ارسال می‌کند. عامل هنگام دریافت یک پیام آلوده ابتدا بررسی می‌کند که آیا نسبت به این تهدید، آسیب‌پذیر است یا خیر، در صورت آسیب‌پذیری احتمال آلودگی بررسی شده و سپس آلوده می‌شود. همچنین عامل بعد از فرا رسیدن موعد تلاش برای ترمیم (به فرض استفاده از مدل SIR)، ابتدا بررسی می‌کند که در وضعیت آلودگی قرار داشته باشد و سپس در صورت موافقت تابع احتمال ترمیم، تغییر وضعیت می‌دهد. برای روشن شدن موضوع الگوریتم این سه رویه در قالب قطعه کدهای (۱) آورده شده است.

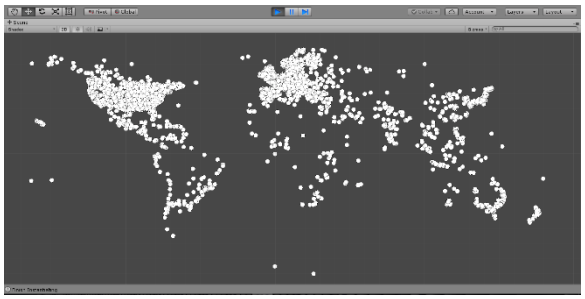
```
void GetMessage(string message){
  int vulnerability =message.GetVulnerability();
  if (state == State.Suspicious){
    if (agent.Is_This_Harmfull_ForMe(vulnerability)){
      bool getInfected =
      CheckInfectionProbability(getInfectedProablity);
      if (getInfected){
        state = State.Infected;
          vulnerabilityId = vulnerability;}}}
  else{
    // Already Infected or Recovered}
}

void TryToRecoverFunction(){
  if (state == State.Infected){
    if (CheckRecoverProbability(getRecoverProablity)){
      state = State.Recovered;}}
}
void TryToMakeConnectionFunction(){
  if (state == State.Infected){
    int victimId = agent.GetAVictimAgentID();
    agent.SendToOtherNode(vulnerabilityId,
    victimId,agentId);
    }
}
int GetAVictimAgentID(){
  int aa = GetARandomConnectedAgent()
  return aa;
}
```

قطعه کد (۱): سه الگوریتم پایه‌ای در هر عامل

۶-۴- نمایش اطلاعات

این بخش همان‌طور که قبلاً تشریح شده است، وظیفه نمایش عامل‌ها، یال‌های اتصالی و آمارهای تحلیلی را بر عهده دارد. بعد از دریافت پیام با هدف مقداردهی اولیه از طرف گره مرکزی، کار این سامانه شروع گشته و پردازش را آغاز می‌کند. این قسمت به‌طور کامل در موتور بازی‌سازی یونیتی پیاده‌سازی شده است.

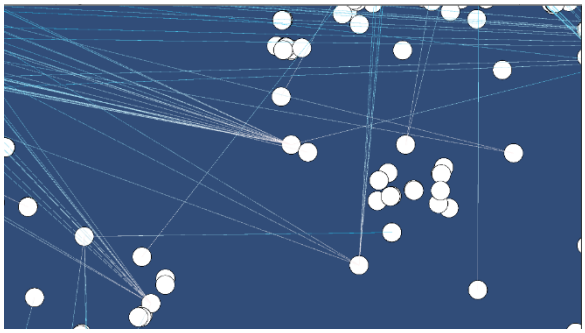
قدم اول بعد از دریافت اطلاعات لازم، تشکیل ساختمان داده مناسب و ساخت عامل‌ها به کمک آن است. در شکل (۹) عامل‌های تولیدشده به‌ازای مجموعه داده مورد آزمایش، را می‌توان مشاهده نمود. در این صحنه ۵۱۴۰۶ عامل وجود دارد و همان‌طور که مشخص است آرایش مکانی عامل‌ها دقیقاً مانند نقشه کره زمین است. دلیل این موضوع واقعی بودن داده‌های مجموعه است. زیرا مجموعه داده بر اساس موقعیت مکانی افراد عضو یک شبکه اجتماعی بوده است.



شکل (۹): عامل‌های مورد آزمایش و نمایش آن‌ها در گره نماسازی

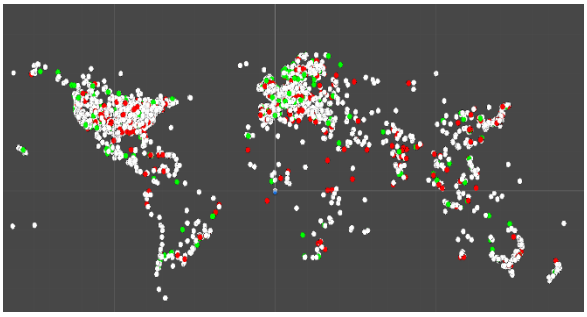
بعد از تکمیل تولید عامل‌ها، نوبت به یال‌های گراف می‌رسد. با توجه به ماهیت مجموعه داده‌های مورد استفاده، گراف ایجادشده در حقیقت یک گراف طبیعی و به طور رسمی یک شبکه بی‌مقیاس است. اما به دلیل مسائل فنی و کارایی و به دلیل تعداد بسیار بالای آن‌ها (حدود ۳۰۰ هزار) مجموعه یال‌ها به این صورت پیاده‌سازی شده‌اند که تنها با کلیک بر روی آن‌ها یال‌های آن‌ها نمایش داده می‌شود. در شکل (۱۰) نمونه‌ای از این موضوع مشخص است.

با اجرای شبیه‌سازی، عامل‌ها از طریق گره‌های محاسباتی تغییر حالت‌های خود را برای این بخش ارسال می‌کنند. گره نماسازی با دریافت این پیام‌ها وضعیت تصویرسازی برای هر عامل را تنظیم می‌کند و با تغییر رنگ آن‌ها کاربر را متوجه وضعیت انتشار می‌کند.



شکل (۱۰): نمایش یال‌ها برای هر عامل در موتور نماسازی

همچنین تعدادی از پارامترهای آماری نیز در این قسمت آورده شده است. پارامترهای «میانگین زمان آلودگی» و «میانگین زمان ترمیم» از جمله این پارامترها است. در شکل (۱۱) نمایی از اجرای شبیه‌سازی بعد از چند دقیقه اجرا آورده شده است.



شکل (۱۱): وضعیت عامل‌ها هنگام شبیه‌سازی

۷- ارزیابی روش

بعد از طراحی معماری مورد نظر و سپس پیاده‌سازی آن، نوبت به آن رسیده است که سامانه نهایی را از ابعاد مختلف مورد آزمایش و بررسی قرار دهیم.

۷-۱- ارزیابی نتایج شبیه‌سازی

معیاری که در این قسمت باید بررسی شود صحت عملکرد شبیه‌سازی و بررسی نتایج آماری به دست آمده است. در این آزمایش‌ها تعداد عامل‌های آلوده در ابتدای کار ۲۰ درصد کل عامل‌ها هستند. لازم به ذکر است، احتمال نهایی آلودگی یک عامل حاصل ضرب دو احتمال آلوده شدن و داشتن آسیب‌پذیری است.

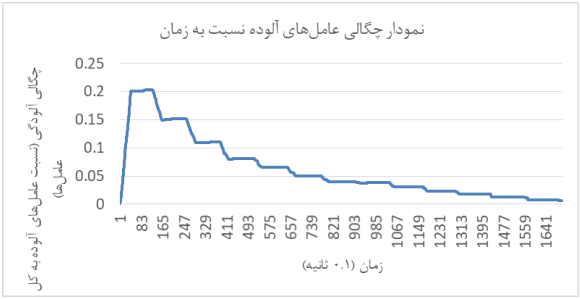
- **آزمایش اول**

هدف از این آزمایش بررسی روال کلی اجرای شبیه‌سازی و بررسی چگالی آلودگی با گذشت زمان است. پارامترهای این آزمایش در جدول (۲) آورده شده است.

جدول (۲): پارامترهای شبیه‌سازی در آزمایش اول

نام پارامتر	مقدار	نام پارامتر	مقدار
زمان تلاش برای ارتباط	۱۰ ثانیه	احتمال آلوده‌شدن	٪۷۵
زمان تلاش برای ترمیم	۱۵ ثانیه	احتمال داشتن آسیب‌پذیری	٪۳۶
مدل انتشاری	SIR	احتمال ترمیم شدن	٪۲۵

همان‌طور که در شکل (۱۲) مشخص است، در ابتدای فرایند شبیه‌سازی، چگالی آلودگی به‌شدت افزایش یافته و شیب تند صعودی دارد. دلیل این جریان، وجود آلودگی‌های ابتدایی در داده‌های اولیه است. این مقدار همان ۲۰ درصد است که در بالا ذکر شد. بعد از این مقداردهی اولیه، و تا زمان ۱۴۳ دهم ثانیه چگالی آلودگی با شیب ملایم رو به افزایش است. دلیل این اتفاق این است که تعداد زیادی عامل آلوده سعی در آلوده‌سازی عامل‌های دیگر دارند. از آنجایی که زمان تلاش برای ترمیم ۵ ثانیه بیشتر از زمان تلاش برای آلودگی است، در نتیجه در تصویر، نمودار به شکل پله‌ای کم می‌شود. در زمان‌های ۱۴۳ و یا ۳۵۶ نزول شدید، کاملاً مشخص است. دقیقاً در این زمان‌ها عامل‌ها سعی در ترمیم خود داشته و با توجه به احتمال، ۲۵ درصد از آن‌ها موفق به ترمیم خود خواهند شد. بار دیگر تأکید می‌کنیم که عامل آلودگی، تنها مقدار احتمال آلودگی نیست، بلکه وجود آسیب‌پذیری در عامل مقصد نیز مهم است. در حقیقت ابتدا وجود آسیب‌پذیری بررسی شده و اگر آسیب‌پذیر بود، احتمال بررسی می‌شود. لذا با توجه به پارامترهای این شبیه‌سازی حاصل ضرب ۰/۷۵ و ۰/۳۶ که معادل ۰/۲۷ است، احتمال آلودگی نهایی است.



شکل (۱۲): چگالی آلودگی با گذشت زمان در آزمایش اول

- **آزمایش دوم**

با توجه به تغییرات لحاظ شده، در جدول (۳) احتمال آلودگی نهایی تا ۹۰ درصد افزایش یافته است بنابراین، توقعی که می‌رود افزایش شیب‌های آلودگی در نمودار است.

همان‌طور که در شکل (۱۳) مشخص است، بر خلاف آزمایش قبلی که شیب‌های صعودی بسیار کم و نامحسوس بوده است، در این آزمایش شیب صعودی بسیار تند است.

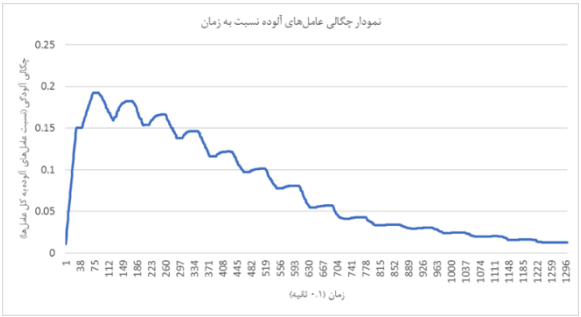
جدول (۳): پارامترهای شبیه‌سازی در آزمایش دوم

نام پارامتر	مقدار	نام پارامتر	مقدار
زمان تلاش برای ایجاد ارتباط	۵ ثانیه	احتمال آلوده‌شدن	٪۹۵
زمان تلاش برای ترمیم	۱۰ ثانیه	احتمال داشتن آسیب‌پذیری	٪۹۵
مدل انتشاری	SIR	احتمال ترمیم شدن	٪۲۵

مخصوصاً در اوایل آزمایش. دلیل این موضوع احتمال آلودگی بسیار بالا است. نکته دیگر این است که در اوایل کار، قسمت‌هایی به شکل تپه دیده می‌شود. این تپه‌ها در حقیقت یک نمودار تصاعدی بوده که در اواخر آن به شکل ثابت در آمده است، و اندکی بعد، نوبت به تلاش عامل‌ها برای ترمیم می‌رسد و یک پله کاهشی را خواهیم داشت. هر چه به سمت اواخر آزمایش نزدیک‌تر می‌شویم، اختلاف طبقات کمتر شده و تغییرات کمتر است.

- **آزمایش سوم**

در این آزمایش، کمی الگوریتم زمان‌بندی برای هر عامل را تغییر داده تا پله‌هایی که در نمودار دیده می‌شود کمتر شود. بر این اساس کافی است با یک تابع توزیع تصادفی یکنواخت این زمان را به طور تصادفی طوری بین عامل‌ها تقسیم کنیم، که میانگین آن همان پارامتر مورد نظر باشد. پارامترهای آزمایش ششم در جدول (۴) آورده شده است.



شکل (۱۳): چگالی آلودگی با گذشت زمان در آزمایش دوم

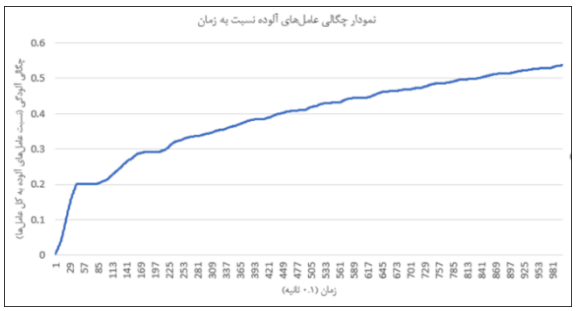
همان‌طور که در نمودار شکل (۱۴) مشخص است، دیگر پله‌های بزرگ با شیب زیاد وجود ندارد و تقریباً نمودار یک مسیر صعودی ثابت را نشان می‌دهد. دلیل این پدیده تصادفی بودن «زمان تلاش برای ارتباط» و «زمان تلاش برای ترمیم» برای هر عامل است. همچنین به دلیل این‌که احتمال ترمیم صفر است، چگالی آلودگی به طور دائم در حال افزایش است.

در این آزمایش هر عامل یک زمان مخصوص به خود را دارد. هرچند که با توجه توضیحات ذکرشده، میانگین این مقدار همان مقدار مشخص‌شده در جدول (۴) است.

جدول (۴): پارامترهای شبیه‌سازی در آزمایش سوم

نام پارامتر	مقدار	نام پارامتر	مقدار
زمان تلاش برای ایجاد ارتباط	۱۰ ثانیه	احتمال آلوده‌شدن	٪۹۵
زمان تلاش برای ترمیم	۱۰ ثانیه	احتمال داشتن آسیب‌پذیری	٪۹۵
مدل انتشاری	SI	احتمال ترمیم شدن	۰

نکته دیگری که وجود دارد کم شدن شیب صعودی با گذشت زمان است. دلیل این پدیده این است که هر چه رو به جلو می‌رویم احتمال انتخاب عامل تکراری که از قبل آلوده شده باشد، بیشتر می‌شود.



شکل (۱۴): چگالی آلودگی با گذشت زمان در آزمایش سوم

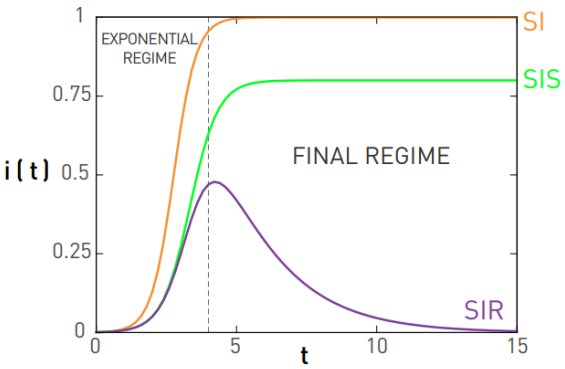
- **ارزیابی نتایج شبیه‌سازی با مدل تحلیلی**

در این قسمت به بررسی صحت نتایج شبیه‌سازی‌ها می‌پردازیم. یکی از روش‌های بررسی این موضوع مقایسه نتایج به‌دست‌آمده با نتایج حاصل از حل تحلیلی و مدل‌های ریاضی می‌باشد. هر چند در مدل‌های تحلیلی همان‌طور که پیش‌تر گفته شده بود، تنها می‌توان به رفتار کلی و ساده‌شده سامانه دست پیدا کرد. بدیهی است که نتایج شبیه‌سازی واقعی‌تر بوده و همین موضوع عاملی برای وجود نویز و بی‌نظمی در نتایج است. به عبارت دیگر نتایج مدل‌های تحلیلی بسیار پیوسته و دارای شکل منظم است.

باراباسی و همکارانش [۱] در فصل دهم کتاب بسیار معتبر "علم شبکه" به بررسی مدل‌های پدیده‌های همه‌گیر می‌پردازد. در این کتاب حل تحلیلی برای مدل‌های SI، SIR و SIS ارائه شده است که شکل (۱۵) به آن اشاره می‌کند. در این شکل رفتار سامانه بر اساس این سه مدل مورد مقایسه قرار گرفته است.

حال با مقایسه بین نتایج شبیه‌سازی روش ارائه شده با کتاب مرجع می‌توان تا حدودی به صحت نتایج پی برد. همان‌طور که در شکل (۱۵) مشاهده می‌شود، نمودار مدل SIR تا حد قابل قبولی با نتایج آزمایش اول و دوم شباهت دارد و آزمایش سوم نیز با نمودار مدل SI تطبیق دارد.

برای تحلیل این نمودارها کافیست که به مدل آن دقت کنیم. در مدل SI تمامی عامل‌ها در ابتدا در وضعیت "مستعد آلودگی" قرار دارند بنابراین، چگالی آلودگی صفر است. با گذشت زمان و در نهایت، هر کدام از این عامل‌ها به حالت "آلوده شده" تبدیل شده و چگالی آلودگی یک می‌شود. این مهم در حالت SIS دچار تغییر می‌شود و سامانه در یک چگالی آلودگی خاص به حالت تعادل می‌رسد. در این حالت نرخ آلوده شدن با نرخ مستعد آلودگی شدن برابر می‌شود. در مدل SIR هم بعد از رسیدن به اوج آلودگی کم‌کم با ترمیم شدن عامل‌ها، نرخ آلودگی به سمت صفر میل خواهد کرد.



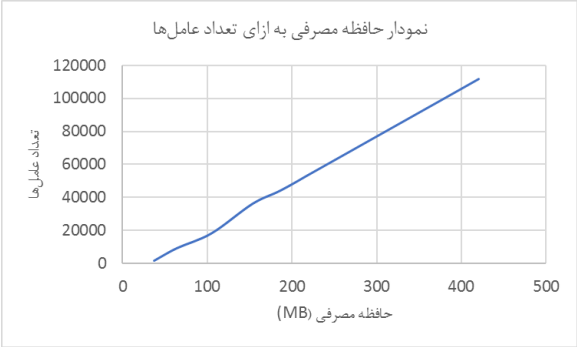
شکل (۱۵): چگالی آلودگی در گذر زمان بر اساس مدل تحلیلی [۱]

۷-۲- تحلیل مقیاس‌وسیع بودن و مقیاس‌پذیری سامانه

یکی دیگر از اهداف معماری ارائه‌شده مقیاس‌وسیع بودن و مقیاس‌پذیر بودن است. این‌که سامانه بتواند تعداد زیادی عامل را شبیه‌سازی کرده و با افزایش آن‌ها نیز بتواند راه حلی ارائه دهد.

آزمایش اول مطرح‌شده در بخش ۷ را از دیدگاه کارایی بررسی خواهیم کرد. بعد از آزمایش‌های به عمل‌آمده مهم‌ترین معیار برنامه، مقدار حافظه مصرفی است. در شکل (۱۶) مقدار حافظه مصرفی گره محاسباتی به ازای تعداد مختلف عامل، نشان داده شده است. آنچه مشخص است، رابطه خطی بین تعداد عامل‌ها و حافظه مصرفی است که از قبل نیز قابل پیش‌بینی بوده است. با توجه به شکل اگر عامل‌ها به تعداد ۱۲۰ هزار باشند، حدود ۵۰۰ MB حافظه مصرف خواهد شد. به همین دلیل با افزایش عامل‌ها ممکن است دیگر یک رایانه قابلیت محاسبه و مدیریت حجم مورد نیاز را نداشته باشد. به همین دلیل است که گره‌های محاسباتی باید به‌صورت توزیع شده باشد. یعنی به تعداد لازم بر روی رایانه‌های مختلف اجرا شده و بار کاری میان آن‌ها تقسیم شود. در همین مثال اگر از ۴ محاسبه‌گر استفاده می‌کردیم، با فرض تقسیم مساوی بار میان آن‌ها، حافظه مصرفی برای هر یک در حالت ۱۲۰ هزار عامل، حدود ۱۰۰ MB خواهد بود و این مقداری ایده‌آل است. از طرف دیگر ممکن است با افزایش عامل‌ها، پردازنده نیز به‌عنوان یک تنگنا محسوب شود که با همین روش توزیع عامل‌ها میان گره‌های مختلف این مشکل حل خواهد شد.

اگر بخواهیم مطلب فوق را به شکل رسمی بیان کنیم کافیست نسبت تعداد عامل‌ها به تعداد گره‌های محاسباتی را در یک بازه خاص ثابت نگه داریم، با توجه به رابطه (۲) در این صورت مقدار حافظه مصرفی به‌ازای هر گره محاسباتی نیز بین یک بازه خاص خواهد بود و می‌توان آن را ثابت در نظر داشت.



شکل (۱۶): رابطه حافظه مصرفی و تعداد عامل‌ها

در رابطه (۲) مقدار حافظه کلی که برای شبیه‌سازی مورد نیاز است قابل مشاهده است، از حاصل ضرب تعداد عامل‌ها در مقدار حافظه مورد نیاز هر یک به‌علاوه یک مقدار ثابت محاسبه می‌گردد.

$$Total\_Ram\_Required = Agents\_Counts \times Agent\_RAM\_Required + Constant \quad (۲)$$

$$Compute\_Node\_Ram\_Required = \frac{Total\_RAM\_Required}{Compute\_Node\_Count} \quad (۳)$$

در رابطه (۳) مقدار حافظه مورد نیاز برای یک گره محاسباتی که در فرایند شبیه‌سازی قرار است شرکت کند، محاسبه می‌شود. همان‌طور که اشاره شد مقدار حافظه مورد نیاز با تعداد گره‌های محاسباتی رابطه خطی معکوس دارد و اگر نسبت حافظه مورد نیاز کلی به تعداد گره‌های محاسباتی ثابت باشد، مقدار حافظه مورد نیاز برای یک گره محاسباتی نیز ثابت خواهد بود. با در نظر گرفتن مطالب بیان شده و با توجه به استفاده از روش‌های زیر، می‌توان طرح پیشنهادی را مقیاس‌پذیر دانست.

- عدم استفاده از معماری متمرکز
- استفاده از گره‌های محاسباتی توزیع شده
- توزیع بار محاسباتی بین گره‌های محاسباتی

- امکان اضافه کردن گره‌های محاسباتی جدید
- امکان استقرار هر یک از گره‌های محاسباتی، گره مرکزی، پردازه سرور و پردازه نماسازی بر روی یک کامپیوتر مجزا
- استفاده از معماری نظیر به نظیر در ارتباط گره‌ها
- پیش‌پردازش فرایندهای مبتنی بر IO و عدم استفاده از دیسک سخت هنگام شبیه‌سازی
- جداسازی بخش تصویرسازی از هسته اصلی برنامه

طبق آزمایشها و مشاهدات صورت گرفته در این فصل، طرح ساخته‌شده آزمایش با تعداد ۶۰ هزار عامل و ۳۰۰ هزار یال ارتباطی را در سامانه آزمایشگاهی که مشخصات آن در جدول (۱) آورده شده است اجرا کرده است و بر این اساس می‌توان نتایج این بخش را رضایت آمیز دانست.

۸- نتیجه‌گیری

در گام اول پژوهش، سعی در طراحی یک معماری نرم‌افزاری داشتیم که بتواند پاسخ‌گوی نیازمندی‌های مورد نظر باشد. برای این کار معماری مذکور در سطوح مختلف تشریح شده و در هر سطح مؤلفه‌های تشکیل‌دهنده و نحوه ارتباط آن‌ها با یکدیگر بررسی شد.

سپس برای ارزیابی معماری اقدام به ساخت و پیاده‌سازی یک محیط شبیه‌سازی برای آزمون و اجرای انواع شبیه‌سازی‌ها زده و این محیط دقیقاً بر اساس معماری یادشده ساخته شده است. با انجام آزمایش‌های مختلف، سامانه از نقطه نظرات مختلف مورد تحلیل و بررسی قرار گرفت. در این مرحله نه‌تنها اطلاعاتی در مورد مسائل فنی سامانه و معماری به دست آمد، بلکه اطلاعاتی در مورد موضوع انتشار از جمله رابطه نرخ احتمال آلودگی و نرخ احتمال ترمیم در سرعت و وسعت انتشار حاصل شد که در جای خود حائز اهمیت است. برای بررسی صحت نتایج پژوهش، نتایج شبیه‌سازی‌ها با نمودارهای حاصل از حل تحلیلی مورد مقایسه گرفت و تطابق قابل قبولی مشاهده شد.

اجرای این شبیه‌سازی‌ها در اندازه‌ای نسبتاً بزرگ و تحلیل و پیش‌بینی رفتار سامانه و در حقیقت معماری ارائه شده در مقیاس وسیع‌تر مشخص کننده این موضوع است که معماری مطرح‌شده یک معماری مقیاس‌وسیع و مقیاس‌پذیر است. یعنی امکان کار با تعداد زیادی عامل را داشته و همچنین با افزایش تعداد عامل‌ها، معماری این قابلیت را به طور بالقوه دارد که بتواند این بار اضافه‌شده را مدیریت کند. همچنین تغییرات به وجود آمده در مدل‌های انتشاری نیز به بدنه اصلی شبیه‌سازی لطمه‌ای وارد نمی‌کند. نمایش اطلاعات آماری و وضعیت عامل‌ها به شکل پویا دیگر امکان این سامانه است که اطلاعات بسیار سودمندی را در اختیار محقق می‌گذارد.

در پایان باید صادقانه به این موضوع اشاره داشت که با وجود دست‌آوردهای زیاد و ویژگی‌های مثبتی که این معماری و سامانه پیاده‌سازی شده دارد، طرح‌های ارائه‌شده خالی از اشکال نبوده و تعدادی مشکل و کاستی در آن وجود خواهد داشت. برخی از این مشکلات هم‌اکنون مشخص هستند و ممکن است مسائل دیگری در ادامه به‌وجود آید.

در هر صورت باید توجه داشت که نقاط ضعف این سامانه با کمی تغییر در نحوه پیاده‌سازی و شاید کمی تغییر در معماری قابل حل است. از جمله این مشکلات می‌توان به مشکل کارایی در بخش نماسازی اشاره کرد که با وجود بررسی‌ها و تلاش‌های زیاد در این زمینه باز هم جای خالی یک سامانه کاملاً مقیاس‌پذیر در این بخش احساس می‌شود. البته باید توجه داشت که آیا اساساً نمایش وضعیت عامل‌ها در این شبیه‌سازی تنها جنبه ادراکی و حسی داشته و یا در استخراج نتیجه هم کاربرد دارد. در صورتی که برای کاربرد مورد نظر این موضوع اهمیت داشته باشد باید راه حلی برای این مسئله پیدا شود. همچنین در مورد گره مربوط به سرور نیز اگر بتوان رویکرد مرکزی را به رویکرد توزیعی و یا غیر مرکزی تغییر داد این قسمت نیز از نظر کارایی بهبود پیدا می‌کند. بنابراین، در ادامه به شکل کلی به مواردی که در آینده می‌توان روی آن‌ها کار کرد اشاره می‌شود:

- راه حلی برای مدیریت در شبکه و نحوه ارتباط عامل‌ها به یکدیگر: به طوری که دیگر این ارتباطات فقط از طریق فایل ورودی قابل تنظیم نباشد. قابلیت انتخاب انواع شبکه‌ها از جمله شبکه مقیاس‌آزاد و یا شبکه تصادفی می‌تواند حائز اهمیت باشد.
- افزایش جزئیات در قسمت آسیب‌پذیری‌ها: در حال حاضر این اطلاعات تنها به‌صورت تصادفی و در هر بار اجرا تنظیم می‌شود. امکان وارد کردن آسیب‌پذیری‌های مختلف از یک پایگاه داده می‌تواند ویژگی خوبی باشد.
- حرکت شبکه، حرکت عامل و تغییر ارتباط عامل‌ها: در حال حاضر تمامی این مؤلفه‌ها به شکل ثابت هستند. اگر بتواند این جزئیات را وارد چرخه شبیه‌سازی کرد، دقت آن به مراتب بیشتر خواهد شد.
- خصوصیت و ویژگی منحصر به فرد برای هر عامل: در حال حاضر تمامی عامل‌ها یکسان هستند و تنها در موقعیت جغرافیایی، وضعیت آلودگی و ارتباطی با یکدیگر تفاوت دارند. اگر هر عامل با توجه به مسائل مختلف خصوصیات خاص‌تری داشته باشد، شبیه‌سازی پیچیده‌تر خواهد شد.

به‌طور مثال خاموش و روشن بودن عامل‌ها، ساعت کاری عامل‌ها، نحوه ارتباط عامل‌ها و غیره، همچنین اختصاصی بودن پارامترهای شبیه‌سازی برای هر یک از عامل‌ها نیز ایده جالبی است. این‌که بعضی از عامل‌ها تمایل به ارتباط بیشتری دارند در حالی که بعضی دیگر، کمتر تمایل به ارتباط دارند.

۹- منابع

[1] A. Barabási and M. Pósfai, "Network Science," Cambridge University Press, 2017.

[2] U. Wilensky and W. Rand, "An introduction to agent-based modelling," MIT Press, 2015.

[3] W. Dubitzky, K. Kurowski, and B. Schott, "Large-Scale Computing Techniques for Complex System Simulations," Wiley, 2011.

[4] M. Ashtiani and M. Abdollahi Azgomi, "A distributed simulation framework for modeling cyber attacks and the evaluation of security measures," Simulation, vol. 90, no. 9, pp. 1071-1102 , 2014.

[5] W. V. D. Broeck, C. Gioannini, B. Gonçalves, M. Quaggiotto, V. Colizza, and A. Vespignani, "The GLEaMviz computational tool, a publicly available software to explore realistic epidemic spreading scenarios at the global scale," BMC Infect Dis, vol. 11, no. 1, 2011.

[6] "NetLogo Home Page," http://ccl.northwestern.edu/netlogo/index.shtml. [Accessed: 09- Dec- 2018].

[7] "NetLogo Models Library: Virus," http://ccl.northwestern.edu/netlogo/models/Virus. [Accessed: 09- Dec- 2018].

[8] S. Hosseini and M. Abdollahi Azgomi, "Malware Propagation Modeling Considering Software Diversity Approach in Weighted Scale-Free Networks," Journal of Electronic and Cyber Defence (ECDJ), 2018. (in Persian)

[9] "What is RabbitMQ?" https://www.cloudamqp.com/img/blog/exchanges-bidings-routing-keys.png. [Accessed: 06- Nov- 2018].

[10] "SNAP: Stanford Network Analysis Project," https://snap.stanford.edu/. [Accessed: 09- Sep- 2017].

[11] Techterms, "Malware Definition," https://techterms.com/definition/malware. [Accessed: 22- Sep- 2017].

[12] R. Leszczyna, I. Nai Fovino, and M. Masera, "Simulating malware with MAlSim," Journal in Computer Virology, vol. 6, no. 1, pp. 65-75, 2008.

[13] J. Krishnaswamy and S. J. S. University, "Wormulator: Simulator for Rapidly Spreading Malware," http://scholarworks.sjsu.edu/etd_projects/69/. [Accessed: 24- 10-2016].

[14] J. Aycock, H. Crawford, and R. Degraaf, "Spamulator: the Internet on a laptop," Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education - ITiCSE '08, pp. 142-147 , 2008.

[15] "GleamVis: The global epidemic and mobility model," http://www.gleamviz.org/simulator/. [Accessed 23 10 2016].

[16] S. A. Thomas, "Data Visualization with JavaScript," O'Reilly, 2015.

[17] S. Yu, G. Gu, A. Barnawi, S. Guo, and I. Stojmenovic, "Malware Propagation in Large-Scale Networks," IEEE Transactions on Knowledge and Data Engineering, vol. 27, no. 1, pp. 170 - 179, 2015.

[18] G. Yan, G. Chen, S. Eidenbenz, and N. Li, "Malware propagation in online social networks: nature, dynamics, and defense implications," ASIACCS '11 Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, pp. 196-206 , 2011.

[19] N. Santhi, G. Yan, and S. Eidenbenz, "CyberSim: Geographic, temporal, and organizational dynamics of malware propagation," Simulation Conference (WSC), pp. 2876 - 2887, 2010.

[20] S. Paul, "Malware Propagation and Detection," LAP Lambert, 2016.

[21] V. Karyotis and M. Khouzani, "Malware Diffusion Models for Modern Complex Networks: Theory and Applications," Morgan Kaufmann, 2016.

[22] M. Garetto, W. Gong, and D. Towsley, "Modeling Malware Spreading Dynamics," IEEE INFOCOM, vol. 3, pp. 1869 - 1879, 2003.

[23] C. Fleizach and M. Liljenstam, "Can you infect me now?: malware propagation in mobile phone networks," WORM '07 Proceedings of the 2007 ACM workshop on Recurring malcode, pp. 61-68 , 2007.

[24] M. R. Faghani and H. Saidi, "Malware Propagation in Online Social Networks," Malicious and Unwanted Software (MALWARE), 4th International Conference, pp. 8 -14, 2009.

[25] D. Chen , L. Wang, and J. Chen, "Large-Scale Simulation: Models, Algorithms, and Applications," CRC Press, 2012.

[25] F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, P. Merson, R. Nord, and J. Stafford, "Documenting Software Architectures," Addison-Wesley Professional, 2011.

[26] P. N. Castillo, "Mastering D3.js - Data Visualization for JavaScript Developers," Packt Publishing, 2014.

[27] A. Bondi, "Characteristics of scalability and their impact on performance," Proceedings of the second international workshop on Software and performance - WOSP '00, 2000.

[28] K. Channakeshava, D. Chafekar, K. Bisset, and M. Marathe, "EpiNet: A Simulation Framework to Study the Spread of Malware in Wireless Networks," Proceedings of the 2nd International Conference on Simulation Tools and Techniques for Communications, Networks and Systems, SimuTools, 2009.

[29] S. Goyal, "Centralized vs Decentralized vs Distributed," Medium, https://medium.com/@bbc4468/centralized-vs-decentralized-vs-distributed-41d92d463868. [Accessed: 06- Oct- 2017].

[30] M. Lees, B. Logan, and A. J. King, "HLA Simulation of Agent-Based Bacterial Models," in Simulation Interoperability Standards Organisation, Genoa, 2007.

[31] O. Topçu, U. Durak, H. Oğuztüzün, and L. Yilmaz, "Distributed Simulation: A Model Driven Engineering Approach. Springer," 2016.

[32] K. Channakeshava, K. Bisset, V. A. Kumar, M. Marathe, and S. Yardi, "High Performance Scalable and Expressive Modeling Environment to Study Mobile Malware in Large Dynamic Networks," IEEE International Parallel & Distributed Processing Symposium, 2011.

[33] N. DuPaul, "Common Malware Types: Cybersecurity 101," https://www.veracode.com/blog/2012/10/common-malware-types-cybersecurity-101. [Accessed: 24- Sep- 2017].

[34] M. Damshenas, A. Dehghantanha, and R. Mahmoud, "A Survey on Malware Propagation, Analysis, and Detection," International Journal of Cyber-Security and Digital Forensics (IJCSDF), vol. 2, no. 4, pp. 10-29, 2013.

Journal of Electronical & Cyber Defence
Vol. 7, No. 2, 2019, Serial No. 26

A Software System for Large-Scale Simulation of Malware Propagation in Computer Networks

E. Ghanad Tavakoli, M. Abdollahi Azgomi*

*Iran University of Scince and Technology

(Received: 02/05/2018, Accepted: 13/10/2018)

ABSTRACT

Today, the presence of methods and tools for large-scale modeling of different types of epidemic phenomena is a serious and critical requirement. Understanding these phenomena and discovering strategic solutions in various situations is also very important. Regarding the large number of agents and their complicated behaviors and lack of theoretical and analytical solutions, the need for simulation systems is so clear. In this paper, we introduce a software system for simulation of epidemic phenomena, especially malware propagation in computer networks. This software system is able to simulate a large number of agents with different propagation models and parameters and show the graphical and statistical results dynamically. Distributed architecture, using real-world data, ability to use different kinds of epidemic models and dynamic visualization of simulation results, are some of the innovations of the proposed method. By performing different simulations, the relationships between infection density and simulation parameters such as the propagation model, the infection probability and the probability of vulnerability, were investigated over time. Additionally, the correctness of the simulation results is confirmed by comparing them to the results obtained from analytical methods.

Keywords: Epidemic Phenomena Modeling, Malware Propagation, Simulation Environment, Distributed Simulation

* Corresponding Author Email: azgomi@iust.ac.ir